Procedia Computer Science

Volume 29, 2014, Pages 2127–2136

ICCS 2014. 14th International Conference on Computational Science

Cyclic hybrid flow-shop scheduling problem with machine setups

Wojciech Bożejko[1], Łukasz Gniewkowski[1],
Jarosław Pempera[1], and Mieczysław Wodecki[2]

[1] Institute of Computer Engineering, Control and Robotics, Wrocław University of Technology
Janiszewskiego 11-17, 50-372 Wrocław, Poland
wojciech.bozejko@pwr.edu.pl
jaroslaw.pempera@pwr.edu.pl
lukasz.gniewkowski@pwr.edu.pl

[2] Institute of Computer Science, University of Wrocław
Joliot-Curie 15, 50-383 Wrocław, Poland
mieczyslaw.wodecki@ii.uni.wroc.pl

Abstract

In this paper we consider an *NP-hard* hybrid flow shop problem with machine setups and cycle-time minimization. The above issue is an important generalization of a flow-shop problem with minimization of a cycle time, and it stays in a direct relationship with a flexible job shop problem. In the hybrid problem task operations are performed by machines arranged in slots, i.e., a set of machines with the same functional properties. In this work we presented a graph model, properties of the problem and methods of determining approximate value of the optimal cycle duration. The above mentioned concepts have been used in the construction of tabu search algorithm. Computational experiments were conducted on well-known in literature examples, which confirmed high efficiency of the algorithm.

Keywords: discrete optimization, metaheuristics, cyclic scheduling.

1 Introduction

For many years, one could observe an increasing market demand for diversity (multiassortment) of production. This may be provided, among many other issues, by cyclic production. In fixed intervals of time (cycle time) there is produced a certain 'batch' (a mix of kit, a set) of assortments. Process optimization is typically reduced to minimization of cycle time. Proper selection of mix and cycle time allows us not only to meet the demand, but also to improve the efficacy and effectiveness of the machinery use. Thus, recently, one can observe a significant increase of interest in the problems of cyclic tasks scheduling theory. For they are usually important and difficult (mostly *NP-hard*) problems from the standpoint of not only theory but also practice.

Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2014

doi:10.1016/j.procs.2014.05.197

A comprehensive overview of the state of knowledge concerning the cyclical task scheduling problem can be found in the work of Levner et al. [12]; analyzing the issues of computational complexity of algorithms for solving various types of cycle scheduling problem. The authors consider in particular NP-difficult problems of various cyclic types including a variety of criterion functions and additional constraints (*no wait, no buffer*, etc.).

In the scientific work by Panwalkar et al. [14] on task scheduling it was found that 75% of problems occurring in practice requires at least one setup dependent on the order of tasks execution. However, in 15% of the problems a setup of all tasks should be taken into consideration. Nevertheless, in the vast majority of works in the field of scheduling setups are not taken into account at all. This applies both to single and and multi-machine problems and to different goal functions. This type of issues are important from both theoretical and practical standpoint.

Continuous flow production systems are among most commonly encountered in industry. Everywhere where the production process consists of the following successive stages, we deal with such systems. Each stage of production is realized in a separate slot supplied with specialized machinery. Every slot is equipped with one or more machines. If in at least one slot there are two or more machines, then we deal with a flow-shop system with parallel machines. In the literature, there are also other names describing this problem such as: a hybrid flow-shop system or a flexible production line. The hybrid flow-shop problem with setups was considered, among many others, in the works [5], [6],[11], [8], [10] and [15]. Also solutions for parallel computing environments are proposed [3, 4].

The work is divided into four main parts. In the first part the problem of cyclic scheduling in a hybrid flow-shop system was formulated and described. In the second part, a graph model and a number of properties used in the effective determining of the cycle time and cyclic scheduling were proposed. In the third part there is proposed an algorithm based on tabu search method. In addition, this chapter presents the elimination property and a property that allows in time $O(1)$ for designation of the cycle time for any solution from the neighborhood. The results of experimental test of an algorithm are shown in the last part.

2 Problem description

The considered in this work hybrid flow-shop problem with machine setups can be formulated as follows:

PROBLEM: There is a given *set of tasks* $\mathcal{J} = \{1, 2, \ldots, n\}$, which should be performed **repeatedly** on machines from the set $\mathcal{M} = \{1, 2, \ldots, m\}$. There is a break up of the set of machines into *types* (*slots*) i.e. subsets of machines with the same functional properties. Each task "passes" through all the slots (technological line). The task is a sequence of *operations*, from which each must be done in a fixed time, on exactly one machine for each nest. The problem consists in allocation of tasks to machines and the determination of the sequence of their execution on machines to minimize the *cycle time*. There must be the following constraints fulfilled:

(i) each task can be performed simultaneously on one suitable type of machine,
(ii) no machine can perform more than one task at the same time,
(iii) task's performance cannot be interrupted,
(iv) a technological line of tasks performance must be preserved,

(v) between successively performed on the same machine, tasks there should be a setup taken into consideration,

(vi) each task is sequentially executed after cycle time is finished.

This problem will be denoted by **CH** in short.

The set of tasks in a single cycle is called MPS (*minimal part set*) in short. MPSs are therefore performed cyclically one after another.

The set of machines $\mathcal{M} = \{1, 2, \ldots, m\}$ can be broken up into m subsets of machines of the same type (*nests*) $\mathcal{C} = \{1, 2, \ldots, c\}$, whereas the i-th ($i = 1, 2, \ldots, c$) type includes m_i machines, which are indexed by the numbers $r_i + 1, \ldots, r_i + m_i$, where $r_i = \sum_{s=1}^{i-1} m_s$, ($r_{=}0$). We denote by $\mu_k \in \mathcal{C}$ a slot, in a which there is a machine $k \in \mathcal{M}$.

Any task must be performed on exactly one machine from each slot. By p_{jz} we denote the time of the task $j \in \mathcal{J}$ performance on any machine from the slot $z \in \mathcal{C}$. The machine must undergo a setup between successive performance of tasks. Let s_{ijk} be a setup time of a machine from the slot $k \in \mathcal{C}$ between task $i \in \mathcal{J}$, and immediately afterwards executed task $j \in \mathcal{J}$. By $\mathcal{J}_k$ we denote the set of tasks assigned to machine $k \in \mathcal{M}$. A sequence of sets of tasks

$$D = [\mathcal{J}_1, \mathcal{J}_2, \ldots, \mathcal{J}_m]$$

such that

$$\mathcal{J} = \bigcup_{k=r_z+1}^{r_z+m_z} \mathcal{J}_k \text{ and } \mathcal{J}_k \cap \mathcal{J}_l = \emptyset,\ k \neq l,\ k, l = r_z + 1, \ldots, r_z + m_z,\ z \in \mathcal{C},$$

we call *allocation of jobs to machines* (in short – *allocation*). By $\mathcal{Q}$ we denote the set of all such assignments.

For a fixed allocation of $D \in \mathcal{Q}$ the order of tasks performance on k-th machine can be represented by permutation $\pi_k = (\pi_k(1), \ldots, \pi_k(n_k))$ tasks from the set $\mathcal{J}_k$, where $n_k = |\mathcal{J}_k|$, whereas the order of tasks performance on all machines π by concatenating these permutations, i.e. $\pi = (\pi_1, \ldots, \pi_m)$. By Φ we denote the set of all such orders.

In the present problem the task set $\mathcal{J}$ should be performed periodically. Let π be an order of performing tasks on the machines. By $S^l_{j,z}$ we denote the starting point for the task $j \in \mathcal{J}$ from the l-th cycle ($l = 1, 2, ...$) on a machine from the slot $z \in \mathcal{C}$ These moments are the solution of the **CH** problem, if the following constraints are fulfilled:

$$S^l_{j,z} \geq S^l_{j,z-1} + p_{j,z-1} \qquad j = 1, \ldots, n,\ \ z = 2, \ldots, c,\ \ l = 1, \ldots, \tag{1}$$

$$S^l_{\pi_k(s),\mu_k} \geq S^l_{\pi_k(s-1),\mu_k} + p_{\pi_k(s-1),\mu_k} + s_{\pi_k(s-1),\pi_k(s),\mu_k} \qquad \begin{array}{l} s = 2, \ldots, n_k, \\ k = 1, \ldots, m,\ \ l = 1, \ldots. \end{array} \tag{2}$$

$$S^{l+1}_{\pi_k(1),\mu_k} \geq S^l_{\pi_k(n_k),\mu_k} + p_{\pi_k(n_k),\mu_k} + s_{\pi_k(n_k),\pi_k(1),\mu_k} \qquad k = 1, \ldots, m,\ \ l = 1, \ldots. \tag{3}$$

$$S^l_{j,z} = S^{l-1}_{j,z} + T \qquad j = 1, \ldots, n,\ z = 1, \ldots, c,\ l = 2, \ldots, \tag{4}$$

where T is a *cycle length*.

Inequality (1) models technological limitations, what is more, it means that the starting point of the task j in z-th slot can begin after the completion of this task in the $z - 1$. slot. On the other hand, inequality (2) represents machine limitations and it means that the task $\pi_k(j)$ executed as j-th on machine in the slot k can begin after completion of the previous task on the machine, i.e. task $\pi_k(j - 1)$. It is important to take into account the machine setup time $s_{\pi_k(s-1),\pi_k(s),\mu_k}$. For any cycle, inequality (3) establishes the relationship between the

last task of the cycle and the first of the next cycle. Last (4) inequality characteristic for the cyclic production determines the relationship between the starting times of tasks on machines in successive MPSs.

Optimization of the process (of the problem**CH**) boils down to designation of the assignment of tasks to machines, and the tasks execution order on machines to minimize the cycle time. For a fixed allocation of tasks to machines, by $T(\pi)$ we denote the shortest cycle time of the order $\pi \in \Phi$. It is important to designate an optimal order $\pi^* \in \Phi$, such that

$$T(\pi^*) = \min\{T(\pi) : \pi \in \Phi\}. \tag{5}$$

Figure 1 shows a cyclic schedule in the form of Gannt graph. This chart includes two more performances of a set of tasks (two MPSs). Tasks execution schedule for the second set is distinguished in color gray. The setup times are shown in bold line.

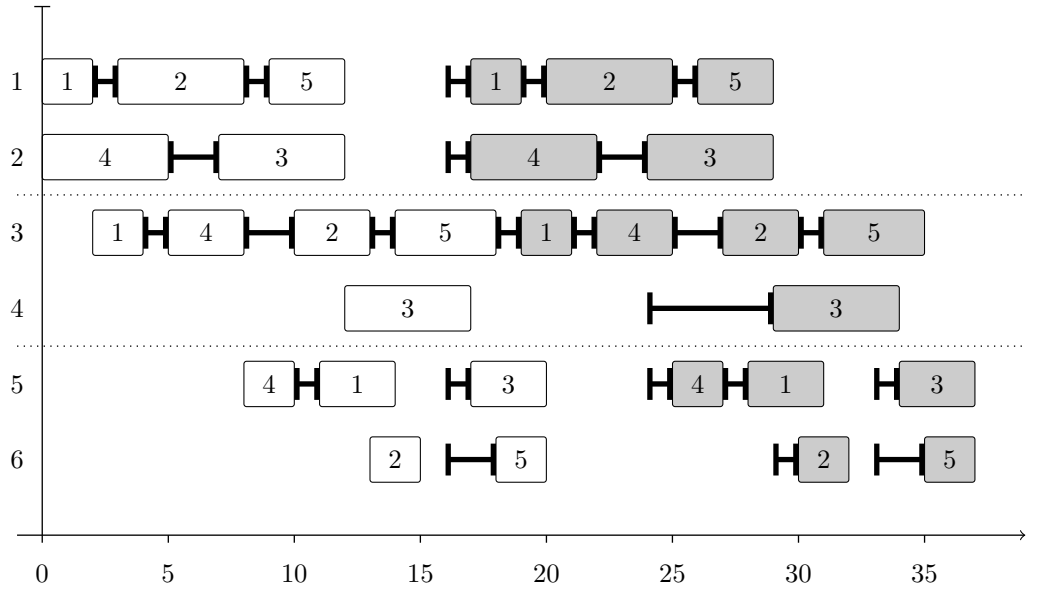

Figure 1: Gantt chart – two successive executions of the jobs set.

3 Properties of the problem

Without loss of generality, in this chapter, we limit ourselves to two first MPSs. Therefore, we are considering a problem which consists in execution of $2n$ tasks from the set $\mathcal{J}^e = \{1, \ldots, n, n+1, \ldots, 2n\}$. Task $n+j$ is a repetition of the task $j \in \mathcal{J}$. For an order $\pi \in \Phi$ of tasks execution from $\mathcal{J}$ we define the so called *extended order* π^e of tasks performance from the set $\mathcal{J}^e$. For a fixed machine $k \in \mathcal{M}$ and order $\pi_k = (\pi_k(1), \ldots, \pi_k(n_k))$, the extended order of tasks execution is

$$\pi_k^e = (\pi_k(1), \ldots, \pi_k(n_k), n + \pi_k(1), \ldots, n + \pi_k(n_k)).$$

For an extended order of tasks execution on machines π^e we define a directed graph with burdened vertices and arcs $G(\pi^e) = (\mathcal{N}, \mathcal{W} \cup \mathcal{E}(\pi^e))$. A set of vertices $\mathcal{N} = \mathcal{J}^e \times \mathcal{C}$. Each vertex

represents the operation of performing the task in the slot. A vertex $(j,k) \in \mathcal{N}$ represents the task $j \in \mathcal{J}$ performed in the slot $k \in \mathcal{C}$. This vertex has a weight p_{j,μ_k} for $j \leq n$ and p_{j-n,μ_k} for $j > n$. A set of arcs $\mathcal{W} \cup \mathcal{E}(\pi^e)$ is a sum:

1. Arcs representing the technological limitations

$$\mathcal{W} = \bigcup_{j=1}^{2n} \bigcup_{z=2}^{c} ((j,z-1),(j,z)).$$

2. Arcs representing machine constraints

$$\mathcal{E}(\pi^e) = \bigcup_{k=1}^{m} \bigcup_{j=2}^{2n_k} ((\pi_k^e(j-1),\mu_k),(\pi_k^e(j),\mu_k)).$$

Arc $((i,k),(j,k)) \in \mathcal{E}(\pi^e)$ has weight s_{i,j,μ_k}. Other arcs have weight 0.

The sequence of vertices of the graph $G(\pi^e)$, from which every two adjacent ones form an arc, *is called a path*. Let $u((x,k),(y,l)) = ((x,k),\ldots,(y,l))$ be the path connecting vertices (x,k) and (y,l). *length* of the path (the sum of weights of vertices and arcs) is denoted by $L((x,k),(y,l))$.

To each vertex $(x,k) \in \mathcal{N}$ we assign number S_{xk} - the starting point of operation $x \in \mathcal{J}^e$ on a certain machine in a slot $k \in \mathcal{C}$. It is easy to show that if in a graph $G(\pi^e)$ there is a path $u((x,k),(y,l)),$, then the moment

$$S_{yl} \geq S_{xk} + L((x,k),(y,l)) - p_{yk}. \tag{6}$$

Theorem 1. *The cycle time satisfies the condition*

$$T(\pi) \geq L((j,z),(n+j,z)) - p_{jz}, \;\; j \in \mathcal{J},\; z \in \mathcal{C}. \tag{7}$$

Proof. A pair of nodes $(j,z)7$ and $(n+j,z)$, $j \in \mathcal{J}$, $z \in \mathcal{C}$ represents a pair of corresponding nodes in two consecutive MPSs. Using the inequality (6)

$$S_{n+j,z} \geq S_{j,z} + L((j,z),(n+j,z)) - p_{jz}, \;\; j \in \mathcal{J},\; z \in \mathcal{C}, \tag{8}$$

then

$$T(\pi) \geq S_{n+j,z} - S_{j,z} \geq L((j,z),(n+j,z)) - p_{jz}, \;\; j \in \mathcal{J},\; z \in \mathcal{C}. \tag{9}$$

■

Theorem 2. *For a fixed order $\pi \in \Phi$ the cycle time*

$$T(\pi) = \max_{k \in \mathcal{M}} L((\pi_k^e(1),\mu_k),(n+\pi_k^e(1),\mu_k)) - p_{\pi_k^e(1),\mu_k}. \tag{10}$$

Proof. It is enough to show that for any order π^e there can be constructed a cyclic schedule with a cycle length $T(\pi)$.

Let $\tilde{G}(\pi^e)$ be a graph constructed from $G(\pi^e)$ by adding certain arcs to it. For each pair of vertices (j,z) and $(n+j,z)$, $j \in \mathcal{J}$, $z \in \mathcal{C}$ we add two additional arcs:

- $((j,z),(n+j,z))$ with weigh $T(\pi) - p_{jz}$, and
- $((n+j,z),(j,z))$ with weigh $T(\pi) - p_{jz}$.

Similarly, in the case of the graph $G(\pi^e)$, to each vertex (j,z) we assign the starting point S_{jz} of execution of task $j \in \mathcal{J}$ in the slot $z \in \mathcal{C}$. The first of arcs $((j,z),(n+j,z))$ generates the constraint

$$S_{n+j,z} \geq S_{jz} + p_{jz} + T(\pi) - p_{jz}, \quad j = 1,\ldots,n,\ z = 1,\ldots,c, \tag{11}$$

whereas the second arc $((n+j,z),(j,z))$ generates the constraint

$$S_{jz} \geq S_{n+j,z} + p_{jz} - T(\pi) - p_{jz}, \quad j = 1,\ldots,n,\ z = 1,\ldots,c. \tag{12}$$

After simple transformations of (12) we obtain

$$S_{jz} + T(\pi) \geq S_{n+j,z} \quad j = 1,\ldots,n,\ z = 1,\ldots,c, \tag{13}$$

Inequalities (11) and (13) are fulfilled when $S_{n+j,z} = S_{j,z} + T(\pi)$, $j = 1,\ldots,n,\ z = 1,\ldots,c$. It is equivalent to the fulfillment of a condition of cyclic schedule.

In the graph $\tilde{G}(\pi^e)$ there are cycles of negative length. Therefore, to determine the longest paths (S_{jz}) there can be Bellman-Ford algorithm applied. ■

Machine $k \in \mathcal{M}$ for which there is an equality

$$T(\pi) = L((\pi_k^e(1),\mu_k),(n+\pi_k^e(1),\mu_k)) - p_{\pi_k^e(1),\mu_k}$$

will be called the *critical* machine.

The problem **CH** is at least NP-hard. Therefore, to come to its solution there will be heuristic algorithms used. The efficiency of such algorithms, in particular based on methods of local searches, is increased by elimination properties. They enable fast removal of worse solutions. For many years, for this purpose, there were the so called block elimination properties used with a huge success (Grabowski et al. [9], Bożejko et al. [1], Wodecki [16]).

Let $u((\pi_k^e(1),\mu_k),(\pi_k^e(n_k+1),\mu_k))$ be a path in a graph $G(\pi^e)$ on a machine $k \in \mathcal{M}$ from gniazda $\mu_k \in \mathcal{C}$. Subsequence $u_{st} = ((u_s,\mu_k),\ldots,(u_t,\mu_k))$, $1 \leq s \leq t \leq n_k+1$ of successive elements of this path is called *a block* (in brief it will be denoted by $B_{st} = (u_s,\ldots,u_t)$), if

1. contains at least 4 elements,
2. for any permutation β_k^e, resulting from π_k^e by swapping the order of elements $u_{s+1},\ldots,u_{t-1}$ there is the following condition satisfied

$$L^\beta((\beta_k^e(1),\mu_k),(\beta_k^e(n_k+1),\mu_k)) \geq L((\pi_k^e(1),\mu_k),(\pi_k^e(n_k+1),\mu_k)), \tag{14}$$

where L^β is the length of a path in a graph $G(\beta^e)$.

It is easy prove that B_{st} is the shortest Hamiltonian path from vertex $(\pi_k^e(s),\mu_k)$ to $(\pi_k^e(t),\mu_k)$ comprising a set of vertices from the set $\{(\pi_k^e(s),\mu_k),\ldots,(\pi_k^e(t),\mu_k)\}$.

Theorem 3. *Let $k \in \mathcal{M}$ be the critical machine to the task performance order $\pi \in \Phi$, a $\{B_1,\ldots,B_g\}$, $B_i = B_{s_i t_i} = ((\pi_k^e(s_i),\mu_k),\ldots,(\pi_k^e(t_i),\mu_k))$, $i = 1,\ldots,g$ a sequence of blocks on the machine. If there is an order $\beta \in \Phi$ such that $T(\pi) > T(\beta)$ then:*

- *at least one task $o \in B_i$ precedes the first task from block$(\pi_k^e(s_i))$, $i \in \{1,\ldots,g\}$, or*
- *at least one task $o \in B_i$ is performed after the last task $(\pi_k^e(t_i))$, $i \in \{1,\ldots,g\}$, or*

- *at least one task $o \in B_i$ is performed on a different from the slot $\mu_k \in \mathcal{C}$, or*
- *at least one task, performed on the machine k not belonging to the blocks, is executed on a different position on this machine or on another machine belonging to the slot $\mu_k \in \mathcal{C}$.*

Proof. The proof is similar to the one stated in work [9]. ■

In the graph $G(\pi^e)$ on the critical machine k there exists n_k paths of length equal to the length of the critical path, i.e. $u((\pi^e_k(s), \mu_k), (\pi^e_k(n_k + s), \mu_k))$, $s = 1, \ldots, n_k$. For each of these paths there can be blocks determined. The number of solutions which do not satisfy the conditions of Theorem 3 increases with the number of elements in blocks. Thus, in order to eliminate as many solutions as possible there should be blocks designated, on the critical path, of the greatest number of tasks.

4 Algorithm of determining cycle time

To solve the given problem there was a known tabu search algorithm for classical hybrid flow-shop problem used (Nowicki and Smutnicki [13]). It is one of the fastest algorithms which uses the block properties. In addition, it is susceptible to the parallelization at the level of vector processing supported by modern processors machine instructions, see [6].

Tabu search is one of the most commonly used methods of constructing heuristic algorithms based on searches of the solution space. The main idea comes down to systematic search of, generated in subsequent iterations, neighborhoods. In the following part of this chapter we will focus on the description of the most important elements of the algorithm.

Four $v = (k, x, l, y)$ represents the move of *insert* type in $\pi \in \Phi$ such that the task $\pi_k(x)$ is removed from the position x, $1 \leq x \leq n_k$ in a permutation π_k and then inserted on position y in the permutationπ_l, $k, l \in \mu_k$, $1 \leq y \leq n_l + 1$ dla $l \neq k$ and $1 \leq y \leq n_k$, $y \neq x$ dla $l = k$.

For each position $x = 1, ..., n_k$ in a permutation π_k defining the order of tasks operations on the critical machine $k \in \mathcal{C}$ by $\mathcal{H}_{k,x,l}$ we denote the set of moves depicting the task $\pi_k(x)$ on every position in the permutation defining the tasks execution order on the machine l, $\mu_l = \mu_k$, $\mu_l \in \mathcal{C}$. On the other hand, by $\mathcal{N}_{k,x,l}$ we denote the subset of $\mathcal{H}_{k,x,l}$ satisfying the constraints of Theorem 3. More specifically,

$$\mathcal{H}_{k,x,l} = \bigcup_{y=1}^{n_l+1}\{(k,x,l,y)\} \text{ for } l \neq k, \quad \mathcal{H}_{k,x,l} = \bigcup_{y\neq x,\ y=1}^{n_k}\{(k,x,l,y)\} \text{ for } l = k. \tag{15}$$

The number of moves in the set

$$\mathcal{N} = \bigcup_{x=1}^{n_k} \bigcup_{\mu_l=\mu_k, l\in\mathcal{M}} \mathcal{N}_{k,x,l}$$

is at the level $O(n_k n)$.

Theorem 4. *Let $\pi \in \Phi$ be the order of tasks execution from a set $\mathcal{J}$, and $\beta \in \Phi$ the order obtained from π after the execution of a move $v = (k, x, l, y)$ w π. Then*

$$L^\beta((\beta^e_i(1), \mu_i), (\beta^e_i(n_i + 1), \mu_i)) = L((\pi^e_i(1), \mu_i), (\pi^e_k(n_i + 1), \mu_i)), \quad for \quad i \neq k,\ i \neq l,\ i \in M, \tag{16}$$

whereas

$$L^\beta((\beta^e_i(1), \mu_i), (\beta^e_i(n_i + 1), \mu_i)) \quad dla \quad i \in \{k, l\} \tag{17}$$

can be designated in time $O(1)$.

Proof. Move $v = (k, x, l, y)$ changes the order of elements of the permutation π_k and a permutation π_l. Other permutations and hence the length of the respective paths are not changed.

Execution of the move $v = (k, x, l, y)$ w π can be divided into two phases: removing an element $j = \pi_k^e(x)$ from position x in π_k and inserting this element in the position y w π_l, obviously the machine l and k must belong to the same slot $z \in \mathcal{C}$,, i.e. $z = \mu_k = \mu_l$. As a result, from the path $u((\pi_k^e(1), z), (\pi_k^e(n_k + 1), z))$ we remove a vertex (j, z), and two arcs $((\pi_k^e(x - 1), z), (j, z))$ i $((j, z), (\pi_k^e(x + 1), z))$. In turn, the removed vertex is replaced with an arc $((\pi_k^e(x - 1), z), (\pi_k^e(x + 1), z))$. In turn, after insertion of task j on position y in π_l we get permutation β_l^e, in which the arc $((\beta_l^e(y - 1), z), (\beta_l^e(y + 1), z))$ existing in π_l is replaced by a sequence consisting of the arc $((\beta_l^e(y - 1), z), (j, z))$, vertex (j, z), and the arc $((j, z), (\beta_l^e(y + 1), z))$. Finally, we obtain

$$L^{\beta}((\beta_l^e(1), z), (\beta_l^e(n_l + 1), z)) = L((\pi_l^e(1), z), (\pi_k^e(n_k + 1), z)) - \Delta_1 + \Delta_2 \quad \text{for} \quad l = k, \tag{18}$$

$$\begin{aligned} L^{\beta}((\beta_k^e(1), z), (\beta_k^e(n_k + 1), z)) &= L((\pi_i^e(1), z), (\pi_k^e(n_k + 1), z)) - \Delta_1 \\ & \qquad \text{for} \quad l \neq k, \\ L^{\beta}((\beta_l^e(1), z), (\beta_l^e(n_l + 1), z)) &= L((\pi_l^e(1), z), (\pi_l^e(n_l + 1), z)) + \Delta_2 \end{aligned} \tag{19}$$

where

$$\Delta_1 = s_{\pi_k^e(x-1),j,z} + p_{j,z} + s_{j,\pi_k^e(x+1),z} - s_{\pi_k^e(x-1),\pi_k^e(x+1),z}, \tag{20}$$

$$\Delta_2 = s_{\beta_l^e(y-1),j),z} + p_{j,z} + s_{j,\beta_l^e(y+1),z} - s_{\beta_l^e(x-1),\beta_l^e(x+1),z}, \tag{21}$$

The values in (20) and (21) can be determined in time $O(1)$. With the known values $L((\pi_k^e(1), z), (\pi_k^e(n_k + 1), z))$ and $L((\pi_l^e(1), z), (\pi_l^e(n_l + 1), z))$, time of designation of the length of paths on machines k and l from the equation (18) or (19), for β is $O(1)$. ■

5 Computational experiments

There were computational experiments whose aim was to assess the effectiveness of the modified tabu search algorithm from work [13]. The study was performed in two phases. In the first phase the effect of block properties on the efficiency of the algorithm was tested, whereas in the second - execution time of algorithm for practical applications.

The algorithm was implemented in C++ language in Visual Studio 2005 and was tested on a Compaq 8510w Mobile Workstation PC computer with Intel Core 2 Duo 2.60 GHz operating Microsoft VISTA system.

The calculations were performed on 960 examples for the hybrid flow-shop problem included in the work [15]. Gathered data was divided into 25 groups. Each consists of examples of the same number of tasks and machines. Individual groups differ from one another in the way of generation and/or the number of machines or tasks. Start solutions were set with the use of a simple construction algorithm based on the method of priority scheduling. As a measure of the algorithm's A efficiency there was a relative error of a goal function value assumed for the best solution π^A determined the algorithm in relation to the reference solution π^{ref}. The error

$$PRD(\pi^A) = 100\%(T(\pi^A) - T(\pi^{ref}))/T(\pi^{ref}). \tag{22}$$

The algorithm from work [13] was implemented in two versions: TSF – with a full neighborhood, generated by the moves from the set $\mathcal{H}$ and TSB – with a reduced environment, generated through the moves from $\mathcal{N}$. There were calculations (of TSF and TSB algorithms) performed for the length of the tabu lists from the set $\{11, 12, 13, 14\}$ and 10,000 iterations. For each example, as the reference solution there was the best found solution adopted. The obtained comparative results are shown in Table 1.

Table 1: PRD values for various length of tabu list.

	L=11		L=12		L=13		L=14	
Group	TSB	TSF	TSB	TSF	TSB	TSF	TSB	TSF
n=20,m=2.4	0.83	0.84	0.72	0.81	0.80	0.79	0.74	0.69
n=20,m=8	0.81	0.66	0.64	0.57	0.51	0.59	0.58	0.70
n=50,m=2	0.39	0.44	0.54	0.34	0.54	0.58	0.54	0.61
n=50,m=4	0.57	0.69	0.48	0.57	0.58	0.56	0.54	0.64
n=50,m=8	0.42	0.47	0.53	0.55	0.46	0.52	0.50	0.57
n=80,m=2	0.52	0.49	0.46	0.56	0.50	0.53	0.45	0.52
n=80,m=4.Var	0.43	0.62	0.41	0.36	0.64	0.22	0.51	0.61
n=80,m=4.Con	0.51	0.81	0.74	0.33	1.03	0.23	0.80	1.03
n=80,m=4.50%	0.24	0.28	0.23	0.21	0.19	0.30	0.24	0.26
n=80,m=4.100%	0.50	0.44	0.40	0.51	0.45	0.37	0.32	0.45
n=80,m=4.125%	0.47	0.59	0.45	0.48	0.50	0.77	0.46	0.66
n=80,m=8.25%	0.11	0.17	0.11	0.12	0.10	0.14	0.13	0.13
n=120.m=2.Var	0.25	0.91	0.27	0.96	0.26	0.82	0.33	0.83
n=120.m=2.Con	0.28	0.53	0.21	0.43	0.23	0.30	0.21	0.32
n=120.m=4.25%	0.11	0.41	0.08	0.40	0.08	0.37	0.10	0.27
n=120.m=4.50%	0.16	0.53	0.13	0.58	0.14	0.56	0.16	0.16
n=120.m=4.100%	2.25	2.56	0.41	0.71	0.40	0.68	0.49	0.81
n=120.m=4.125%	0.29	0.87	0.37	0.75	0.32	0.82	0.42	0.42
n=120.m=5.50%.Con	0.12	0.69	0.13	0.17	0.10	0.38	0.13	0.13
n=120.m=8.25%.Var	0.04	0.05	0.03	0.02	0.02	0.03	0.02	0.02
n=120.m=8.25%.Con	0.10	0.73	0.10	0.37	0.10	0.42	0.09	0.45
n=120.m=8.50%.Var	0.10	0.12	0.13	0.14	0.05	0.11	0.07	0.13
n=120.m=8.100%.Var	0.12	0.17	0.17	0.18	0.07	0.16	0.11	0.19
n=120.m=8.100%.Con	0.30	0.50	0.37	0.46	0.20	0.28	0.42	0.38
n=120.m=8.125%.Var	0.20	0.41	0.25	0.23	0.10	0.26	0.19	0.19
n=120.m=8.125%.Con	0.45	0.53	0.59	0.97	0.39	0.59	0.51	0.47
Average	0.41	0.60	0.34	0.45	0.34	0.44	0.35	0.45

The obtained results clearly show that the TSB algorithm is significantly more effective than the TSF one. For each group of data and length of tabu lists, the average relative error of the TSB algorithm is at least by 0.1 % less than of TSF algorithm. The best results of the two algorithms were obtained for a list length $L = 13$. For this length the average error (for all groups of data) of TSB algorithm is at the level of 0.34 %, whereas for TSF algorithm it equals 0.44 %. It is thus higher by about 30%. In addition, the error of the TSB algorithm decreases with an increasing number of tasks. For groups with a number of tasks $n = 20$ it ranges within $0.45 - 1.03\%$ (except for the group: n=80, m=4, 50% n=80, m=8, 25%). In groups of the largest number of tasks it varies from 0.02 to 0.39 %.

6 Summary

The paper considered the cycle flow-shop hybrid problem with machine setup and minimization of the cycle length. The properties which were used in the construction of an algorithm based on tabu search method proved to be successful. Computational experiments were carried out,

which confirmed the high efficiency of the algorithm. The methodology is designed to solve a class of cyclic scheduling problems, difficult problem of discrete optimization. Proposed optimization techniques, as a part of computational science methodology, can be applied in any real-world optimization problem with permutational representation of a solution.

Acknowledgements. The work was supported by the OPUS grant DEC-2012/05/B/ST7/ 00102 of Polish National Centre of Science.

References

[1] Bożejko W., Grabowski J., Wodecki M., Block approach-tabu search algorithm for single machine total weighted tardiness problem, Computers & Industrial Engineering, Elsevier Science Ltd, Volue 50, Issue1/2 (2006), 1–14.

[2] Bożejko W., Pempera J., Wodecki M., Cyclic hybrid flow shop problem benchmark library, http://staff.iiar.pwr.wroc.pl/wojciech.bozejko/benchmarks.htm

[3] Bożejko, On single-walk parallelization of the job shop problem solving algorithms, Computers & Operations Research 39 (2012), 2258–2264.

[4] Bożejko, Solving the flow shop problem by parallel programming, Journal of Parallel and Distributed Computing 69 (2009), 470–481.

[5] Bożejko W., Uchroński, Wodecki M., Parallel hybrid metaheuristics for the flexible job shop problem, Computers & Industrial Engineering, 59 (2010), 323–333.

[6] Bożejko W., Pempera J., Smutnicki C., Parallel Tabu Search Algorithm for the Hybrid Flow Shop Problem, Computers and Industrial Engineering, 65 (2013), 466–474.

[7] Bożejko W., Wodecki M., Parallel genetic algorithm for minimizing total weighted completion time, Lecture Notes in Computer Science No. 3070, 2004, 400–405.

[8] Brucker P., Kampmeyer T., Cyclic job shop scheduling problems with blocking. Annals of Operations Research, 159 (2008), 161–181.

[9] Grabowski J., Skubalska E., Smutnicki C., On Flow Shop Scheduling with Release and Due Dates to Minimize Maximum Lateness, Journal of the Oper. Research Society, 34,7 (1983), 615–620.

[10] Kampmeyer T., Cyclic Scheduling Problems, Ph. D. Thesis, University Osnabrück, 2006.

[11] Kochhar S, Morris R, Wong W., The local search approach to flexible flow line scheduling, Engineering Costs and Production Economics, 14,1 (1988), 25–37.

[12] Levner E., Kats V, Lopez A.P., Cheng T.C.E., Complexity of cyclic scheduling problems: A state-of-the-art survey, Computers and Industrial Engineering, 59 (2010), 352–361.

[13] Nowicki E., Smutnicki C., The flow shop with parallel machines: A tabu search approach. European Journal of Operational Research, 106 (1998), 226–253.

[14] Panwalkar S.S., Dudek R.A., Smith M.L., Sequencing research and the industrial scheduling problem, Symposium on the theory of scheduling and its applications (ed. Elmaghraby S.E.), Springer-Verlag, Berlin, 1973.

[15] Ruiz R., Serifoglu F.S., Urlings T., Modeling realistic hybrid flexible flowshop scheduling problems, Computers and Operations Research, 35 (2008), 1151–1175.

[16] Wodecki M., A block approach to earliness-tardiness scheduling problems, International Journal on Advanced Manufacturing Technology, 40 (2009), 797–807.